TITLE:

連結文字列の圧縮長について (アルゴリズムと計算理論の新展開)

AUTHOR(S):

遊佐, 俊彦

CITATION:

遊佐, 俊彦. 連結文字列の圧縮長について (アルゴリズムと計算理論の新展開). 数理解析研究所講究録 2012, 1799: 187-190

ISSUE DATE:

2012-06

URL:

http://hdl.handle.net/2433/172979

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
第1799巻 2012年 187-190

2011年度冬のLAシンポジウム [S9]

連結文字列の圧縮長について

遊佐 俊彦*

概要

M. Li 等により，背景知識を全く必要としない，Kolmogorov Complexity を元にしたデータ間の類似度を測る汎用な尺度である NID（正規化情報距離）が提案された [4]．しかし，Kolmogorov Complexity は計算不可能であるため，その代用として現実の圧縮アルゴリズムを用いて類似度を測る尺度 NCD（正規化圧縮距離）が提案されている．NCD は，音楽ファイルの分類や文学ファイルの分類・DNA データに対して適用しうまくデータ間の距離を測ることができることが実験で知られている [2] ものの，NCD がなぜ文字列の類似度（距離）をうまく測れるかは知られていない．そこで本稿では，NCD の圧縮アルゴリズムに Burrows-Wheleer Transform（以下 BWT）をもとにしたアルゴリズムを用いた尺度 $\mathrm{NCD}_{\mathrm{BW}}$ について解析を行った．NCD の計算で重要なのは文字列 x と y を連結した文字列 xy の符号長である．本稿では，文字列として独立かつ一様に生成した文字列 x とその文字列中の文字を確率 p で置換した文字列 y というモデルを考える．文字列 xy の符号長を考える上で大切なのは BWT 変換後の構造であり，その構造でも特に文字の切り替え個数が重要であることに着目することにより，このモデルのもとで，文字を置換した割合 p に関して文字置換の個数は $1-(1-p)^3$ の割合で増えていくという結果を得，実験結果ともほぼ合致した．

*東京工業大学 大学院情報理工学研究科

1 背景と準備

1.1 Compression Distance

事実 1 ([4, 2]). K を Kolmogorov Complexity，x, y を文字列としたとき，x, y の NID を以下のように定義する．

$$\mathrm{NID}\,(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}.$$

$K(x)$ は，入力なしで x を出力する最小プログラム長，$K(x|y)$ は入力 y で x を出力する最小プログラム長である．NID は，$K(x) - K(x|y)$ が x に含まれる y の情報量という考え方に基づいて提案されたものである．この NID を考える際問題となるのが，Kolmogorov Complexity が計算不可能という点である．そこで，Kolmogorov Complexity を定義する究極の圧縮アルゴリズムを，現実に使われている圧縮アルゴリズムで近似する正規化圧縮距離が提案された．

事実 2 ([4, 2]). 圧縮アルゴリズム C によって文字列 x を圧縮したときの圧縮長を $\mathrm{L}_C(x)$ としたとき，文字列 x, y の C による NCD を以下のように定義する．

$$\mathrm{NCD}_C(x,y) = \frac{\mathrm{L}_C(xy) - \min\{\mathrm{L}_C(x), \mathrm{L}_C(y)\}}{\max\{\mathrm{L}_C(x), \mathrm{L}_C(y)\}}.$$

この式を考える上で重要なのは分子の第一項の $\mathrm{L}_C(xy)$ である．これは文字列 x の後ろに文字列 y を連結した文字列の符号長であるが，極端な例として x と y が全く同じ文字列だと仮定すると，$\mathrm{L}_C(xy) \approx \mathrm{L}_C(x)$ となり，x と y が全くの独立な文字列だとすると $\mathrm{L}_C(xy) \approx \mathrm{L}_C(x) + \mathrm{L}_C(y)$ と考えられる．つまり，文字列 x, y を連結したときの符号長 $\mathrm{L}_C(xy)$ が文字列 x の後ろに連結する文字列 y の構

造によってどのように増加・減少するかを解析することが，NCDという尺度が文字列の類似度を測れていることを知るのにとても重要なのである．

1.2 Burrows-Wheleer Transform

本稿では，$L_C(xy)$の解析に用いる圧縮アルゴリズムCはBurrows-Wheleer Transform(BWT)[1]を用いたアルゴリズムとした．このアルゴリズムはbzip2などにも用いられている方法で，文字列の順序を入れ替えるアルゴリズムである．例を用いてBWTアルゴリズムを説明する．BWTは入力文字列$x=$kyotoに対して左に1文字ずつシフトすることで，kyoto，yotok，otoky，tokyo，okyotを得る．さらにこれらの文字列をソートすることによりkyoto，okyot，otoky，tokyo，yotokを得るが，それぞれの文字列の語尾の文字を連結した文字列と，入力文字列が何番目にあるかのペア(otyok, 1)が出力となる．今回はBWT後の構造に着目するので1は省略して，$bwt(x)=otyok$と書くことにする．文字列をソートしているので特定の部分文字列の前に出てくる文字は，出力の際にまとまって出力される．実際，BWTを英語のテキストに適用した場合同じ文字が連続して出てくる傾向があることが知られている．

さらに，BWTをベースとした圧縮アルゴリズムでは，Move To Front(以下MTF)と呼ばれる方法[3]で$bwt(x)$を整数へと符号化する．MTFは，今見ている文字が前に出現してから，異なる文字が何文字出現したかを整数で表したものである．例えば，aabdaacdの場合は，00132032となる．同じ文字が連続して出現すれば0がたくさん出現することになる．この得られた整数を，δ符号やγ符号，ハフマン符号で符号化することですることで，入力文字列の符号を得る．

2 連結文字列の符号長

文字列x, yを連結した文字列についての解析を行う．本稿では，文字列として独立かつ一様に生成した文字列xとその文字列中の文字を確率pで置換した文字列yというモデルを考える（文字列の長さはnでアルファベットの種類数はhとする）．これにより確率構造を導入することができ，解析を可能とした．

まずは，後ろに連結する文字列が入力文字列と同じ場合を考える．入力$xx=$acdbacdbに対してBWTを行うことで，$bwt(xx)=$bbddaaccを得る．これをよく見ると同じ文字が2つずつ連続して出現していることがわかる．これは，表（以下BWT表と呼ぶ）

a	c	d	b	a	c	d	b
a	c	d	b	a	c	d	b
b	a	c	d	b	a	c	d
b	a	c	d	b	a	c	d
c	d	b	a	c	d	b	a
c	d	b	a	c	d	b	a
d	b	a	c	d	b	a	c
d	b	a	c	d	b	a	c

表1: 文字列xxのBWT表

からもわかる通り同じ文字列を連結するとシフトしたときも同じ文字列が2つずつ存在するからである（これをpairと呼ぶ）．さらにBWTで得られた文字列$bwt(xx)$をMTFをしたとき，同じ文字が2つ連続して出現しているので，2文字目は必ず0で符号化される．より小さい数字の方が短い符号で符号化されるδ符号やγ符号などでは，0という符号は長さを稼げているということになる．この観察から，pairが崩れずに存在しているほど連結したときの符号化のロスが少ないと考えることができる．pairが崩れないということは，異なる文字に切り替わる個数が$bwt(x)$と比べて増えないということである．以下では，文字列yの構造によって$bwt(xy)$の異なる文字に切り替わる個数がどう変化していくかを解析する．

解析方針は，文字列xxのBWT表を基準にyの構造によって文字列xyのBWT表がどう変化していくかを考えることである．異なる文字に切り替わ

る個数が $bwt(x)$ に比べて増えるのは，2通り考えられる．

(i) 置換されたことにより行が動きpairが崩れる場合

(ii) 行の最後の文字が置換されたことによりpairが崩れる場合

(i)については，1番目の文字が置換されている場合と，2番目の文字が置換されている場合のみを考える．1番目の文字が置換されている場合は，その文字が異なる文字に置換されてかつ，x 中に置換される前の文字以外が存在する場合なので，pairが崩れる確率は，$\left(1-\frac{1}{h}\right)\left(1-\left(1-\frac{1}{h}\right)^{n-1}\right)$ となる．よってpairが崩れる個数は h が十分大きいとき1である．1番目の文字が置換されている行数は np 行あるのでpairが崩れる個数は $1\times np = np$ 個であり，異なる文字に切り替わる個数も np 個である．2番目の文字が置換されている場合も同様に考えるが，1番目の文字が置換されている場合はすでに数えているので，1番目の文字が置換されていない確率 $1-p$ をかけることで異なる文字に切り替わる個数が，$np(1-p)$ 個であることがわかる．

(ii)については，最後（$2n$ 番目）の文字が置換されている場合は必ずpairが崩れているので，異なる文字に切り替わる個数は，$2n$ 番目の文字が置換されている行の個数で np 個である．しかし，(i)ですでに数えている切り替え個数もこの np 個には含まれてしまっているので，その分を引く必要がある．xx のBWT表に存在するpairは n 組で，今，(i)によってpairが $np+np(1-p)=np(2-p)$ 組崩れてしまっている．よって，pairとして存在しているのは $n-np(2-p)$ 組である．これは，pairで存在しているものとそうでないものの割合を考えると二項分布になるので，(i)で数えていてかつ(ii)でも数えている条件である，pairで存在していない行の $2n$ 番目が置換されている期待値は，$\sum_{m=0}^{np} m\binom{np}{m}(p(2-p))^m(1-p(2-p))^{np-m} = np^2(2-p)$ である．

以上(i)(ii)より，以下の結果を得る．

定理 3. 先に定義したモデルで生成した文字列 x, y について，$bwt(x)$ で異なる文字に切り替わる個数を $d(bwt(x))$ とすると，

$$\begin{aligned}&\mathbf{E}\left(d\left(bwt\left(xy\right)\right)-d\left(bwt\left(x\right)\right)\right)\\ =\ & 2np+np(1-p)-np^2(2-p)\\ =\ & n\left(1-(1-p)^3\right)\end{aligned}$$

である．

この式を実験結果のグラフと重ねると以下の図のような結果となる．実験は $n=10,000$，$h=26$ で行ったものである．図を見ると，理論値を表すグラフと実験値を表すグラフがほぼ同じような曲線を表していることがわかる．また，文字置換確率が1に近づくにつれて，独立かつ一様に生成した長さ $2n$ の文字列の切り替え個数に収束していることが見て取れる．これは，置換するにつれて文字列 xy が独立かつ一様に生成した文字列に近づくからであると考えられる．(ただし，グラフの横軸は文字置換の確率，縦軸は切り替えが起きた個数である．グラフは，実線が解析結果，点線が実験結果横軸に平行な直線が一様分布で長さが $2n$ のものの切り替え個数を表している．)

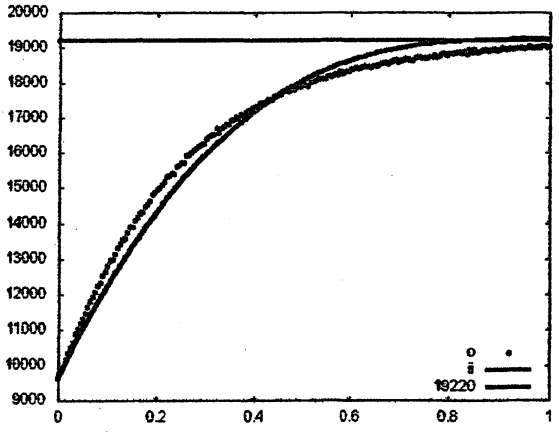

図 1: 実験結果と解析結果

さらに，NCDの圧縮アルゴリズムを異なる文字に切り替わる個数とすると，以下の補題を得る．

補題 4. 先に定義したモデルで生成した文字列 x, y に対して，異なる文字に切り替わる個数 d に関しての NCD_d は，

$$\mathrm{NCD}_d(x,y) = \frac{n\left(1-(1-p)^3\right)}{d(x)}$$

である．

これは，p が 1 に近づくにつれて NCD の値が 1 のに近づき，p が 0 に近づくにつれて NCD の値が 0 に近づいているので，NCD の性質をよく表しているといえる．

3 まとめと今後の課題

本稿では，NCD を解析するために大切な連結文字列の圧縮長について理論的な解析を行った．同じ文字列を連結するという一番簡単で単純なもので，BWT 後の構造を観察し，その構造の中で特に重要そうなものに着目することで，問題をより解析しやすいものにまで変形し，さらに文字列生成に確率構造を導入することにより文字列 x と x を置換した文字列 y について理論解析を行った．その結果，文字置換確率 p について単調に増加するという結果を得ることができた．

今後の課題は，「置換」という操作だけではなく「削除」「挿入」といった操作についての解析を行い，これら三つの操作を組み合わせてできた文字列がどのような符号長を持つか解析することである．

参考文献

[1] M. Burrows and D.J. Wheeler. A block-sorting lossless data compression algorithm. 1994.

[2] R. Cilibrasi and P.M.B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[3] R. Tarjan J. Bentley, D. Sleator and V. Wei. A locally adaptive data compression scheme. *Communications of the ACM*, 29(4), 1986.

[4] M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.